**Delete nodes having greater value on right :-**

Medium Accuracy: 35.51% Submissions: 96K+ Points: 4

Given a singly linked list, remove all the nodes in the list which have any node on their right whose value is greater. (Not just immidiate Right , but entire List on the Right)

**Example 1:**

**Input:**

LinkedList = 12->15->10->11->5->6->2->3

**Output:** 15 11 6 3

**Explanation:** Since, 12, 10, 5 and 2 are

the elements which have greater elements

on the following nodes. So, after deleting

them, the linked list would like be 15,

11, 6, 3.

**Example 2:**

**Input:**

LinkedList = 10->20->30->40->50->60

**Output:** 60  
**Explanation:** All the nodes except the last  
node has a greater value node on its right,  
so all the nodes except the last node must  
be removed.

**Your Task:**  
The task is to complete the function **compute**() which should modify the list as required and return the head of the modified linked list. The **printing**is done by the **driver**code,

**Expected Time Complexity:** O(N)  
**Expected Auxiliary Space:** O(1)

**Constraints:**  
1 ≤ size of linked list ≤ 105  
1 ≤ element of linked list ≤ 105  
**Note**: Try to solve the problem without using any extra space.

**Code :-**

//{ Driver Code Starts

#include<bits/stdc++.h>

using namespace std;

struct Node

{

int data;

Node\* next;

Node(int x){

data = x;

next = NULL;

}

};

void print(Node \*root)

{

Node \*temp = root;

while(temp!=NULL)

{

cout<<temp->data<<" ";

temp=temp->next;

}

}

// } Driver Code Ends

/\*

The structure of linked list is the following

struct Node

{

int data;

Node\* next;

Node(int x){

data = x;

next = NULL;

}

};

\*/

class Solution

{

public:

Node \*compute(Node \*head)

{

Node \*ptr=head, \*before=NULL, \*after=NULL;

// reverse the linked list

while(ptr){

after = ptr;

ptr = ptr->next;

after->next = before;

before = after;

}

ptr = after;

// building own linked list for answer

int maxi = INT\_MIN;

Node \*right=NULL;

while(ptr){

maxi = max(maxi, ptr->data);

if(maxi <= ptr->data){

Node \*left = new Node(maxi);

if(right)

left->next = right;

right = left;

}

ptr = ptr->next;

}

return right;

}

};

//{ Driver Code Starts.

int main()

{

int T;

cin>>T;

while(T--)

{

int K;

cin>>K;

struct Node \*head = NULL;

struct Node \*temp = head;

for(int i=0;i<K;i++){

int data;

cin>>data;

if(head==NULL)

head=temp=new Node(data);

else

{

temp->next = new Node(data);

temp = temp->next;

}

}

Solution ob;

Node \*result = ob.compute(head);

print(result);

cout<<endl;

}

}

// } Driver Code Ends

**T.C :- O(N)**

**S.C :- O(1)**